아날로그 기상 데이터를 OCR로 디지털화할 수 있을까?

글 | 이수경 samantha.lee@kakaobrain.com 2016년 3월 알파고와 이세돌 9단이 펼치는 세기의 대결을 두 눈으로 목도한 이후 인공지능을 제대로 공부해봐야겠다고 결심했습니다. 인공지능 본진이자 연구소인 카카오브레인으로 걸어들어온 이유입니다. 인공지능 기술과 이로 인해 바뀔 미래 사회를 다루는 글을 통해 사람들과 소통하고 싶습니다.

기술감수 | 홍상훈 ian.theman@kakaobrain.com 카카오브레인에서 새로운 아이디어를 고민하고, 이를 직접 구현하는 리서치 엔지니어로 활약 중입니다. 학부 시절부터 인공지능, 뇌 연구에 관심이 많았습니다. 딥러닝의 잠재력에 매일 놀라고 있지만, 완전한 인공지능이 될 때까지는 아직 갈 길이 멀다고 생각합니다. 카카오브레인의 뛰어난 사람들과 함께 AI 기술 발전에 이바지를 하고, 이 기술을 활용해 인류 사회에 보탬이 되고자 합니다.

"OCR 전문가를 찾습니다. 기후 과학자(기상학자)들이 지난 10여 년간 작성한 기후 측정 데이터만 수백만 페이지에 이릅니다. 대부분 사람 손으로 직접 작성했죠. OCR로 과거 기후의 데이터를 인식하고 앞으로 일어날 위험을 예측하는 게 가능할까요?" 영국 레딩 대학(University of Reading)의 기후학자 에드 호킨스(Ed Hawkins)가 2017년 11월 자신의 트위터(Twitter) 계정에 올린 트윗이다.

[그림 1] 에드 호킨스의 트위터[*1]

세월의 흐름에 따라 마모되는 종이로는 기후 데이터를 영구적으로 보존하기가 어렵다. 디지털화하더라도 이미지 형태로 저장하는 것이라 텍스트 원문 검색은 불가능하며 별도의 색인 목록에 의존할 수밖에 없다.[*2] 에드 호킨스가 영구 보존과 내용 검색을 가능하게 하는 기술로 광학 문자 인식(optical character recognition, OCR)을 고려한 이유다. OCR는 인쇄된 문서나 이미지에서 텍스트를 인식하는 기술을 가리킨다.

딥러닝 기반 OCR 파이프라인

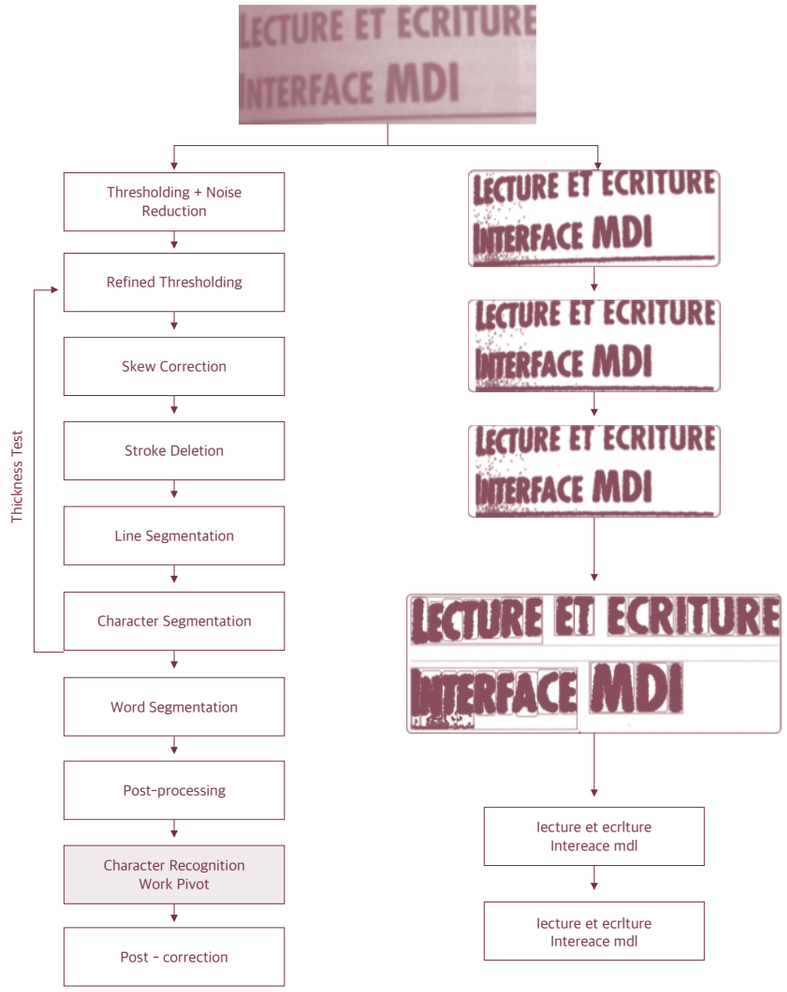
전통적인 OCR 파이프라인(pipeline)의 예는 [그림 2]에서 볼 수 있다. 텍스트 라인(text line)을 찾는 모듈과 문자(letter)를 분할하는 모듈 등 세분된 모듈로 동작한다. 서로 다른 패턴을 인식하는 특징(feature)은 인간 개발자가 직접 설계해야 했다([그림 3]).[*3] 아울러 평판 스캐너로 촬영한 고품질 이미지에서만 제한적으로 동작했다. 인쇄물의 스캔 상태가 좋지 않거나 사람이 직접 작성한 손필기 문서의 인식률이 상대적으로 좋지 못했던 이유다.

*1 참고 | https://twitter.com/ed_hawkins/status/935521005870645248

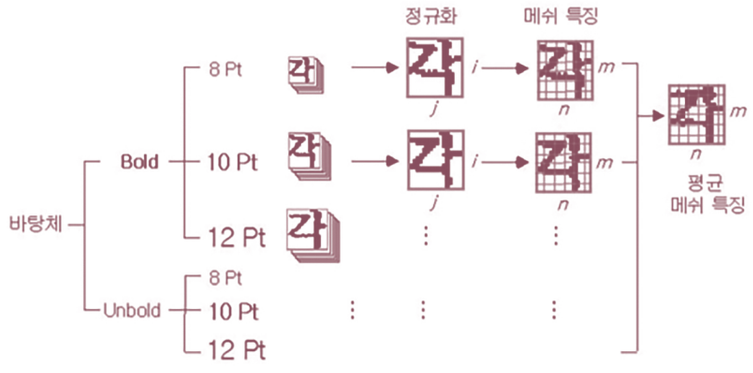
*2 참고 | 김재평, 《인공지능과 기록관리의 미래》, 국가기록원, 2017.

*3 참고 | https://blogs.dropbox.com/tech/2017/04/creating-a-modern-ocr-pipeline-using-computer-vision-and-deep-learning

[그림 2] 전통적인 OCR 파이프라인의 예[*4]



[그림 3] 폰트, 크기, 스타일에 따라 각기 다른 특징을 일일이 정의해야 한다. 게다가 한글은 자모의 조합까지 모두 고려해야 한다[*5]



*4 참고 | C ´eline THILLOU, "Degraded Character Recognition", 2003-2004.

*5 논문 | 박상철 외 2인, <영상 처리 : 문자별 특징 모델을 이용한 한글 문서 영상에서 키워드 검색>, 《정보처리학회논문지B》 12권5호, 정보처리학회, 2005. 10.

*6 참고 | https://medium.com/@rudrasaha1993/paper-summary-see-towards-semi-supervised-end-to-end-scene-text-recognition-7b7efbc9363a

*7 논문 | Xuebo Liu, 'FOTS: Fast Oriented Text Spotting with a Unified Network', 2018.

*8 참고 | https://www.youtube.com/watch?time_continue=3&v=gsZxtO7Unyg

*9 참고 | https://medium.com/@rudrasaha1993/paper-summary-see-towards-semi-supervised-end-to-end-scene-text-recognition-7b7efbc9363a

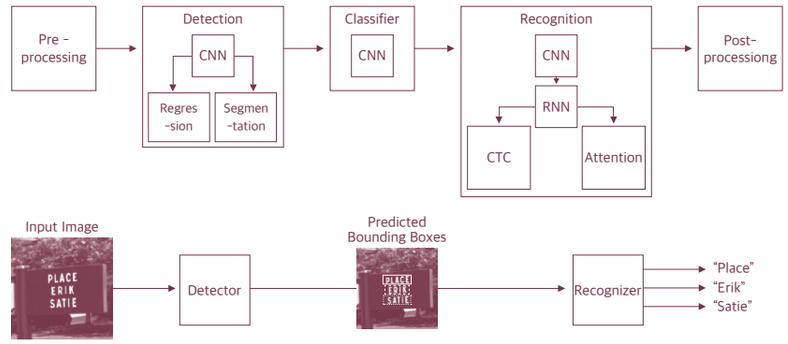
*10 참고 | http://solarisailab.com/archives/303

*11 참고 | https://d2.naver.com/helloworld/8344782

*12 참고 | https://stackoverflow.com/questions/39233823/opencv-for-ocr-how-to-compute-thresholding-levels-for-gray-image-ocr

다행히 컴퓨터 비전(computer vision)에서 이뤄진 기술적 발전으로 OCR 기능 또한 크게 향상됐다. CNN(convolutional neural networks)이 바로 그 주역이다. 대량의 데이터 학습을 통해 이미지에서 텍스트를 인식하는 규칙(feature extraction)을 스스로 만들어낸다. 오늘날의 딥러닝 기반 OCR는 이미지에서 텍스트 영역(text box)을 검출하는 부분과 텍스트 영역의 텍스트를 개별적으로 인식하는 부분 2가지로 나누어볼 수 있다.[*6] 모델에 따라서는 글자 검출 부분과 글자 인식 부분을 단일 모델로 구축해(E2E 방식) 학습 속도나 성능을 높이기도 한다.[*7]

[그림 4] 딥러닝 기반 OCR 파이프라인[*8*9]



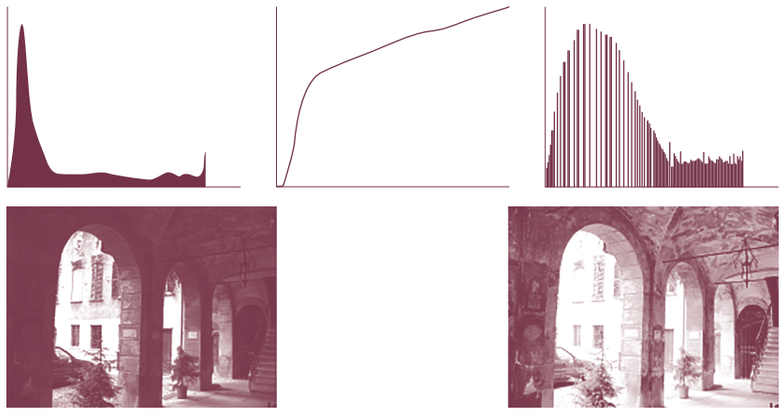
1) 전처리(pre-processing)

엄밀하게 말하면 컴퓨터는 이미지에서 글자와 글자가 아닌 것을 직관적으로 구분하지 못한다. 비슷한 밝기값(intensity)을 가진 픽셀을 덩어리로 인식하는 쪽에 더 가깝다.[*10] 주변과 그 색깔이 다르고 연속된 점의 구조가 다른 글자는 더 쉽게 탐지해낼 수 있다. 이를 두고 한 개발자는 "마치 검은 배경에서 하얀색 덩어리를 찾아 묶는 것과도 같다"고 묘사하기도 했다.[*11]

[그림 5] 컬러 이미지를 회색조로 변환한다[*12]

하지만 문서를 디지털로 처리하는 과정에서는 잡영(noise) 혹은 왜곡(distort)으로 인해 인식률이 현저하게 낮아지는 경우가 많다. 원본 이미지의 밝기, 명암, 왜곡에 따라 인식률의 변화가 매우 심하기 때문이다.[*13] 이에 전처리 과정을 거치면 인식률을 크게 개선할 수 있다.[*14]

[그림 6] 히스토그램을 정규화한다[*15]



[그림 7] 이진화한다[*16]

Input PDF

Here we shall develop an implementation for queues that is based on Pascal pointers. The reader may develop a cursor-based implementation that is analogous, but we have available, in the case of queues, a better array-oriented representation than would be achieved by mimicking pointers with cursors directly. We shall discuss this so-called "circular array" implementation at the end of this section. To proceed with the pointer-based implementation, let us define cells as before:

```
type
    celltype = record
        element: elementtype;
        next: ↑ celltype
    end;
```

Expected output

Here we shall develop an implementation for queues that is based on Pascal pointers. The reader may develop a cursor-based implementation that is analogous, but we have available, in the case of queues, a better array-oriented representation than would be achieved by mimicking pointers with cursors directly. We shall discuss this so-called "circular array" implementation at the end of this section. To proceed with the pointer-based implementation, let us define cells as before:

```
type
    celltype = record
        element: elementtype;
        next: ↑ celltype
    end;
```

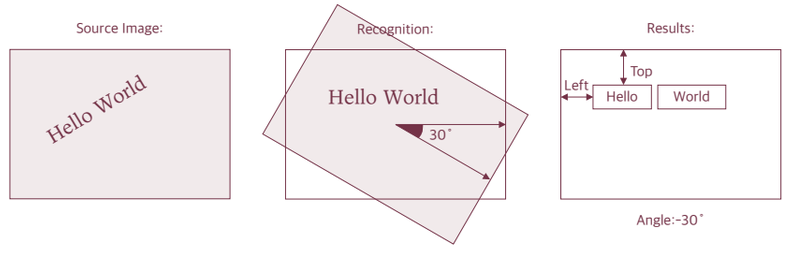
대표적인 전처리 과정은 다음과 같다(딥러닝 파이프라인에 따라서는 일부 과정이 생략될 수 있다). 먼저 컬러 이미지를 회색조(grayscale)로 변환한다([그림 5]). 픽셀값의 범위가 그레이스케일(0~255) 사이에 분포될 수 있도록 이미지를 변형했다는 의미다. 이어 히스토그램 정규화(histogram equalization)를 진행한다([그림 6]).[*13] 영상의 밝기 분포를 재분배해 명암 대비를 최대화하면 더 선명한 이미지를 가질 수 있다. 하지만 배경과 문자를 명확하게 구분하는 데는 여전히 한계가 있다. 이를 해결하기 위해 이진화 작업(binarization)을 진행한다([그림 7]). 픽셀값이 255(흰색)면 '0'으로, 0~254(회색 및 검은색)면 '1'로 바꾸는 것이 골자다. 그 결과, 배경과 글자를 좀 더 명확하게 분리할 수 있다.[*13] 특히 문서 이미지는 흑백 이미지로 변환했을 때 더 좋은 성능을 낸다.[*8] 이 과정에서 생긴 불필요한 잡영을 제거하는 기법이 추가로 도입되기도 한다. 전처리 과정을 거치면 최종적으로 [그림 8]처럼 보다 선명한 이미지를 획득할 수 있다.

[그림 8] 이미지의 전처리 과정 예시[*13]

2) 글자 검출(text detection)

이미지를 CNN에 넣은 후 특징(feature map)을 얻는다. 구하려는 데이터는 텍스트 영역(text box)과 텍스트 영역의 회전 각도(rotation angle)다. 입력 이미지에서 텍스트 영역을 골라내면 불필요한 계산을 줄일 수 있다. 회전 정보는 기울어진 텍스트 영역을 수평 형태로 만들 때 활용한다([그림 9]). 그런 뒤 이미지를 텍스트 단위로 잘라낸다. 이 단계를 거치고 나면 개별 글자(character) 이미지 또는 단어(word) 이미지를 획득할 수 있다([그림 10]).

[그림 9] 텍스트 영역의 회전 각도 정보를 활용해 수평 모양으로 만든다[*17]



[그림 10] 이미지화된 텍스트 분할[*18]

3) 글자 인식(text recognition)

이미지를 잘라내고 난 뒤에는 각 이미지가 어떤 글자를 포함하고 있는지를 인식해야 한다. 이 과정에서도 CNN이 사용된다([그림 11]). CNN은 이미지 형태의 개별 단어·글자를 인식하는 방법을

*13 논문 | Kim Jae-wan, "A Personal Prescription Management System Employing Optical Character Recognition Technique", Journal of the Korea Institute of Information and Communication Engineering, 2015.

*14 논문 | Shin Hyunsub, 'A Study on Performance Improvement of Business Card Recognition in Mobile Environment', Journal of the Korea Institute of Information and Communication Engineering, 2014.

*15 참고 | https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/ch01.html

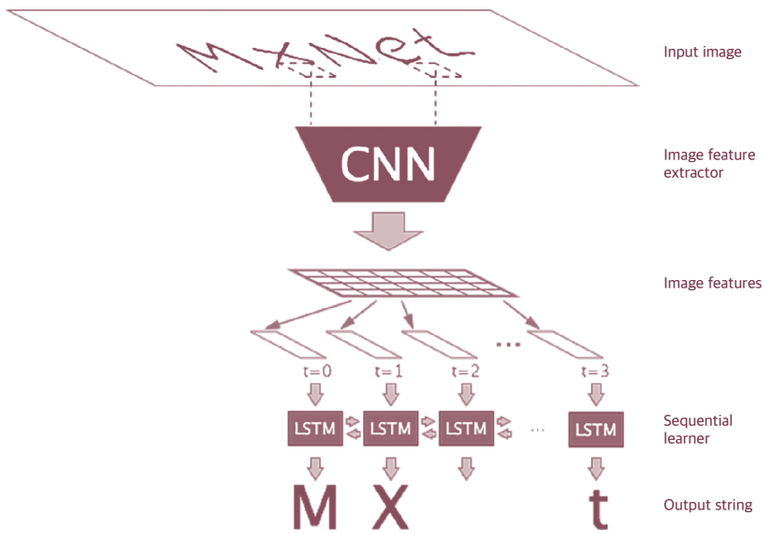
*16 참고 | https://askubuntu.com/questions/396437/how-can-i-remove-the-gray-scale-page-background-of-a-pdf-document-scan-while-pre

*17 참고 | https://docs.microsoft.com/en-us/uwp/api/windows.media.ocr.ocrresult.textangle

*18 참고 | https://en.wikipedia.org/wiki/Automatic_number-plate_recognition#/media/File:California_license_plate_ANPR.png

학습한다. 잘 학습된 CNN이라면 [그림 10]처럼 4, Y, C, H, 4, 2, 8이라는 문자열을 출력한다. 표준 글꼴(font)과 포맷(format)을 갖춘 문자 이미지일수록 글자를 효율적으로 인식할 수 있다. 한편, CNN이 인식할 수 있는 단어나 문자열 종류는 언어마다 다르다. 따라서 범용적인 OCR를 위해서는 이미지만 가지고 언어를 추정하는 모듈도 구비할 필요가 있다.

[그림 11] 글자 인식을 위한 일반적인 프레임워크*19



4) 후처리(post-processing)

[그림 12]에 쓰인 텍스트 문장을 보자. 기계는 이미지에 묻은 먼지로 인해 생긴 얼룩 때문에 알파벳 C를 O로 오인식할 수도 있다. 해당 부위를 들여다보는 것만으로는 내막을 파악할 길이 없다. 반면, 사람은 '귀리(oat)가 매트에 앉았다'가 아니라 '고양이(cat)가 매트에 앉았다'로 읽는다. 문서가 발행된 맥락, 문서 내용의 맥락 등을 파악해 문서를 이해한 덕분이다.

[그림 12] 텍스트 오인식 예제*20

OCR도 사람이 텍스트를 받아들이는 방법과 유사한 방식으로 글자 인식 오류를 후처리한다. 방법으로는 크게 2가지가 있다. 첫 번째는 글자마다의 특징을 이용하는 방법이다. '일/밀', '먕/양', '파/따'처럼 유사한 형태의 글자(유사 쌍)를 구분해 오류를 정정한다.*21 두 번째는 문맥 정보를 이용하는 방법이다. 이를 위해서는 언어 모델이나 사전이 필요한데, 웹상의 수많은 텍스트

*19 참고 | https://medium.com/apache-mxnet/handwriting-ocr-handwriting-recognition-and-language-modeling-with-mxnet-gluon-4c7165788c67

*20 참고 | https://www.hyperscience.com/blogs/data-entry-automation

데이터를 학습한 언어 모델을 딥러닝으로 구축하기도 한다. 예를 들어, '홋길동'이 인식되었다고 보자. 맥락을 보면 이는 이름임을 알 수 있다. '홋'이라는 성은 존재하지 않으므로 입력 오류로 판단할 수 있을 것이다.*14 또는 '아버지가방에들어가신다'처럼 띄어쓰기가 제대로 인식되지 않았다면, 언어 모델을 통해 '아버지가 방에 들어가신다'라는 더 그럴듯한 문장을 추론할 수 있다.

[표 1] 글자 인식 오류*22

대치(substitution)	다른 문자로 인식한다
실종(missing)	문자를 인식하지 못한다
추가(insertion)	없는 문자가 새로 추가된다
조합(combination)	두 문자가 하나로 결합한다
분해(decomposition)	하나의 문자가 두 개로 분해된다

OCR가 완벽한 솔루션이 아닌 이유

OCR는 아날로그 형태로 존재하는 문서를 디지털로 변환하고 문서 속 텍스트를 검색하는 방법으로 널리 활용되고 있다. 물론 기술적으로 100% 완벽하지는 않다. 아직은 매우 한정된 조건에서만 극도로 좋은 결과를 산출해낸다.

1) 디지털화

가장 먼저 디지털화 단계에서부터 큰 난관에 봉착하는 경우가 부지기수다. 종이 형태로 보관된 문서를 디지털 이미지로 변환하는 데는 많은 시간과 비용이 들기 때문이다. 에든버러 대학교(Edinburgh University)가 진행하고 있는 스코틀랜드 법원 자료*23 디지털화 프로젝트(Scottish court of session papers digitization project)를 한 번 살펴보자.*24

[그림 13] 에든버러 대학교의 스코틀랜드 법원 자료 디지털화 프로젝트*24

지난해 초 이뤄진 파일럿 프로젝트에서는 방대한 자료를 효과적으로 디지털화하는 데 주안을 뒀다. 일률적인 방법으로는 디지털화가 불가능한 상황이었는데, 어떤 책의 두께는 18cm를 훌쩍 넘었고 책마다 제본 방식도 제각기 달랐다. 먼저 책마다 스캔 환경을 설정해야만 했다. 책 스캐너에 책을

*21 논문 | 장승익 외 1인, <유사 문자 쌍을 구분하기 위한 한글 인식의 후처리>, 한글 및 한국어 정보처리 학술대회, 2001.

*22 논문 | 안재철, <OCR 소프트웨어를 이용한 한글 문서 검색 시스템>, 전북대학교 석사학위논문, 2002.

*23 설명 | 에든버러 대학교 도서관을 비롯해 여러 소장처가 보유하고 있다. 18~19세기 사이 스코틀랜드의 고등법원에서 나온 법률 기록물로, 대략 1800권으로 구성돼 있다. There are over 5000 volumes made up of written pleadings of contested cases, answers, replies, and case summaries, many of which have contemporary annotations.

*24 참고 | http://libraryblogs.is.ed.ac.uk/diu/2017/03/03/scottish-court-of-session-papers-digitisation-pilot

펼치다가 책등이 망가질 것을 우려해 일부 책은 디지털 카메라로 페이지 단위로 사진을 일일이 찍어야 했다. 책의 상태에 따라서는 디지털로 변환하는 과정에서 더 많은 손상이 가해질 수 있어 일부는 작업에서 제외할 수밖에 없었다. 300DPI 이상의 고해상도로 문서를 스캔할 수 있다는 점에서 이 프로젝트는 그나마 나은 상황에 속한다. 새로 스캔을 떠야 하는 대상이 파손 또는 소실돼 저해상도의 이미지만 남았다면 높은 인식 정확도를 기대하기 어렵기 때문이다.

2) 보편성(색, 글씨체)

손글씨를 디지털로 변환하는 작업은 도전 그 자체다. 글씨를 쓰는 사람 수만큼 존재하는 서로 다른 패턴을 완벽하게 인식하는 일은 불가능에 가깝기 때문이다. 실제로 글을 쓰는 사람이 서로 달라서 생기는 변이(intraclass variation)가 한 사람이 글씨를 쓸 때 생기는 변이보다 큰 것으로 알려졌다.*21 또한, 정자체일수록 인식률이 높고, 필기체일수록 인식률이 낮다. 1850년대 이전 타자기로 작성된 텍스트의 인식 정확도 역시 낮을 수 있다.

[그림 14] 필기체일수록, 글자 크기가 작을수록 OCR 인식이 어렵다. 일정한 크기 이상으로 작성된 정자체 문장은 OCR 인식이 상대적으로 쉽다.*25

흐릿한 색상의 글자나 흐린 회색 바탕에 조금 진한 회색 글자도 쉽게 구분하는 사람과는 달리, 컴퓨터는 명확한 경계(edge)가 있어야만 글자를 구분해낸다. 이런 이유로 사진 속 옥외 간판 텍스트를 인식하는 게 책의 글씨를 인식하기보다 더 어렵다. 고품질 스캐너로 스캔한 책의 OCR 성능이 여권 이미지보다 월등히 높다. 특히 흰색 배경에 검은색 텍스트를 가장 잘 인식한다.

에드 호킨스를 위한 최고의 방법은?

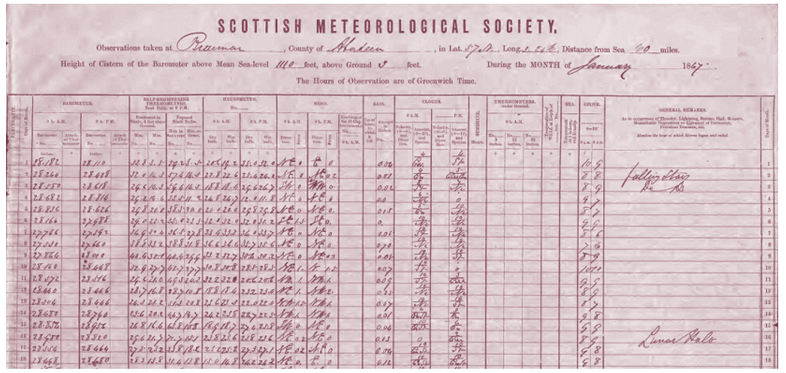
에드 호킨스가 디지털화를 원하는 기상 관측 데이터의 일부를 살펴보자.*26 스캔해야 할 문서의 수가 절대적으로 많은 것은 사실 큰 문제가 아닐 수 있다. 그보다는 손글씨로 쓴 텍스트를 훈련시키는 단계에서 난관에 부딪힐 공산이 크다. 강인한(robust) 텍스트 인식 알고리즘 개발 자체가 불가능하다는 의미는 아니다. 앞서 언급한 대로 필기체일수록, 그리고 여러 사람이 작성한 글씨일수록 학습하는 데에 더 많은 시간이 걸릴 뿐이다. 또한, 표 형태의 문서를 처리하는 데도 기술적인 어려움이 따를 수 있다.*27 정리하자면 에드 호킨스의 디지털화 프로젝트에는 기존의 상용 OCR로는 해결되지 않는 문제가 산재해 있다는 것이다.

*25 참고 | https://files.realpython.com/media/sample3.8d93cef43018.jpg

*26 참고 | https://www.dropbox.com/s/hn26v08benxlotp/1867_Vol_1_Braemar_21.pdf?dl=0

*27 참고 | https://read.transkribus.eu/2018/04/16/help-us-process-tables

[그림 15] 기상학자들이 수기로 작성한 기후 측정 데이터*1



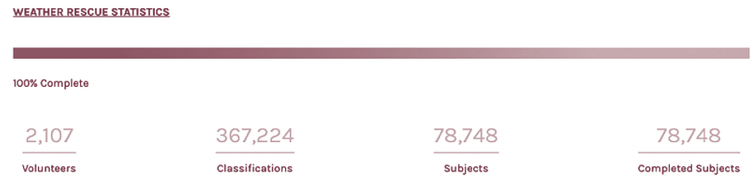
지난 자료를 디지털화하는 데 시간과 비용이 큰 문제로 작용한다면, 디지털화 작업으로 얻을 수 있는 가치에 의문을 제기할 수밖에 없다. 이에 대해 에드 호킨스는 자신의 트위터에서 "디지털화하려는 기후 데이터가 상업적 가치를 지니고 있는지는 확실하지 않다"면서 "다만 박애적인 접근(philanthropic approach)이 필요한 영역인 것 같다"고 말했다. 지난 수십 년간의 기후를 기록한 데이터를 분석하는 부분은 학술적인 관점에서 큰 의의가 있다고 판단했음을 추론해볼 수 있다.

기상 데이터를 스캔해 디지털 파일 형태로 저장하는 작업은 이미 완료했다고 가정하자. 그다음 해야 할 일은 라벨링(labelling)이다. 정답이 라벨링된 데이터를 이용해 학습 모델을 찾고 난 이후에야 완전히 새로 주어진 데이터에서도 답을 스스로 추론해낼 수 있기 때문이다. 딥러닝 성능에 큰 영향을 미치는 요소인 만큼 양질의 데이터를 대규모로 확보하는 게 무엇보다 중요하다.

사람이 일일이 정답을 알려줘야 하기에 이 라벨링 단계에서 대규모 자원이 투입된다. 일반 기업이나 연구실에서는 Mturk*28나 Captricity*29처럼 작업 요청자(requester)와 작업자(worker)를 이어주는 크라우드 소싱(crowd sourcing) 서비스를 많이 활용한다.*30 과제를 가진 요청자가 과제를 서비스에 올리면 전 세계에 작업자로 등록된 사람들이 실시간으로 과제를 수행한다.

상업적인 목적을 띠지 않는 만큼, 시민의 자발적인 참여를 유도하는 플랫폼을 이용하는 방법도 있다. 시민이 참여하는 과학 프로젝트 플랫폼인 주니버스(Zooniverse)가 대표적인 사례다. 한 연구팀은 250여만 장의 암세포 사진을 분석해야 했다. 18개월은 족히 걸리는 작업이었다. 이에 연구팀은 주니버스에 프로젝트를 공개했고, 단 3개월 만에 분석 작업을 끝냈다.*31 에드 호킨스 또한 주니버스에 '웨더레스큐(weatherrescue.org)'*32 프로젝트를 게재한 적도 있다.

[그림 16] 웨더레스큐 프로젝트의 통계자료*32



*28 참고 | https://www.mturk.com

*29 참고 | https://captricity.com

*30 참고 | http://www.venturesquare.net/3045

*31 참고 | http://www.spreadi.org/zooniverse

*32 참고 | http://weatherrescue.org

대량의 라벨링 데이터를 확보한 후에는 딥러닝 기반 OCR 파이프라인을 구축하는 작업이 이뤄져야 할 것이다. 데이터 과학 커뮤니티인 캐글(Kaggle)에 과제를 올려 전 세계 머신러닝 전문가들의 도움을 받는 방법이 대표적인 예다. 에드 호킨스는 자신의 트위터를 통해 “OCR 프로젝트는 엄청난 도전이며, 기계가 아닌 사람이 직접 데이터를 입력하는 게 가장 훌륭한 솔루션임을 알게 됐다”며 “구글(Google)이나 아마존(Amazon), 애플(Apple) 같은 기업의 도움이 필요할 것 같다”고 말했다.

〈아날로그 기상 데이터를 OCR로 디지털화할 수 있을까?〉 브런치로 연결되는 QR 코드입니다.